REMARKS

Applicants respectfully request the Examiner's reconsideration of the present application as amended.

Claims 1-9, 18-22, and 26-30 and are pending in the present application.

Claims 1-9, and 18-22 are rejected under 35 U.S.C. §102(e) as being unpatentable over U.S. Publication Number US 2003/02117250 (Bennett).

Claims 1-9, and 18-22 are rejected under 35 U.S.C. §102(e) as being unpatentable over U.S. Publication Number US 2005/0091022 (Levit-Gurevick).

The specification has been amended at the paragraph starting at line 17, on page 8.

Support for the amendments to the specification is found on pages 8 and 9 and in Figure 3. Applicant submits that no new matter has been added.

Claims 26-30 have been added.

Support for the new claims are found on pages 5-18 of the specification, Figures 1-8 in the drawings, and claims 1-9, and 18-22 as originally filed. No new mater has been added.

The Examiner has rejected claims 1-9, and 18-22 under 35 U.S.C. §102(e) as being unpatentable over Bennett and Levit-Gurevick.

It is submitted that Bennett and Levit-Gurevick do not render claims 1-9, 18-22, and 26-30 unpatentable under 35 U.S.C. §102(e).

Bennett includes a disclosure of a command pertaining to one or more portions of a register being received from guest software. A determination is made as to whether the guest software has access to all of the requested portions of the register based on indicators within a mask field that correspond to the requested portions of the register. If the guest software has access to all of the requested portions of the register, the command received from the guest software is executed on the requested portions of the register (see Bennett Abstract).

Levit-Gurevick includes a disclosure of an apparatus used to simulate a multiple-processor system by creating multiple virtual machines. The multiple virtual machines may be formed within a single central processing unit (CPU) hardware implementing Virtual Machine Extension (VMX) technology. In an example, the apparatus includes a host environment and a virtual environment that includes the multiple virtual machines. Virtual code may be executed on each of the multiple virtual machines under the control of a direct execution monitor within the host environment. The direct execution monitor may create the virtual machines and control exit and entry thereto. The direct execution monitor may monitor the virtual machines for sensitive events that are to be handled by the host environment, not the virtual environment. The direct execution monitor may determine the nature of the sensitive event, such as whether the instructions associated with the sensitive event should be de-virtualized and simulated separately. The apparatus allows the virtual code to operate as though it is operating on its own dedicate physical processor at a native level (see Levit-Gurevick Abstract).

It is submitted that Bennett and Levit-Gurevich do not teach or suggest executing a plurality of input output (IO) instructions from an instruction stream during a single virtualization event.

On the contrary, Bennett only discloses that VM exits are caused by virtualization events. A VM exit may be generated when guest software attempts to perform an operation (e.g., an instruction) that may result in its access of certain privileged hardware resources (e.g., a control register of an IO port) (see Bennett paragraph [0023-0024]). When guest software attempts to access a privileged resource, the control over the processor is transferred to the VMM 112, which then decides whether to perform a requested operation (e.g., emulate it for the guest software, proxy the operation directly to the platform hardware 116, etc.) or deny access to the resource to facilitate security, reliability or other mechanisms. The VMM 122 does not execute a plurality of IO instructions from an instruction stream during a single virtualization event (see Bennett paragraph [0020] and Figure 1).

Furthermore, Levit-Gurevich only discloses that VM exits occur when the VM 206 or the VM 208 attempts to perform some sensitive event (also termed a Virtualization Event), e.g., an instruction or operation to which the attempting virtual machine does not have privileges or access. Virtualization events include hardware interrupts, attempts to change virtual address space (Page Tables), attempts to access I/O devices (e.g., I/O devices (e.g., I/O instructions), attempts to access control registers, and page faults (see Levit-Gurevich paragraph [0024] and Figure 2). Upon a Virtualization Event, a block 324 within the DEX Monitor 212 detects the Virtualization Event and saves the Guest state data. A block 326 determines if the Virtualization Event is a complex event or not. If the Virtualization Event is not complex, then the block 328 checks if the exit Event was due to the simulated processor end of quota. If the block 238 determines that the answer is no, then control is passed to block 330 which executes code to perform a virtualization operation to handle the Virtualization Event with the DEX Monitor 212. The block 330 may perform the simulation needed for handling a non-complex event, which does not have to be sent to the Full Platform Simulator 210. The block 330 then passes control back to the block 314 for Guest state virtualization. If, he answer at the block 328 is yes, then a block 332 switches the DEX Monitor 212 to control of the next Vmx+1. Control is passed to block 314 (see Levit-Gurevich paragraph [0035] and Figure 3). Levit-Gurevich does not teach or suggest executing a plurality of IO instructions from an instruction stream during a single virtualization event.

In contrast, claim 1 states

> A method for performing virtualization, comprising:
> executing a plurality of input output (IO) instructions from an instruction stream during a single virtualization event.

(Claim 1) (Emphasis added).

Claims 18 and 30 include similar limitations. Given that claims 2-9 depend directly or indirectly from claim 1, and claims 19-22, and 26-29 depend directly or indirectly from claim 18,

it is likewise submitted that claims 2-9, 19-22, and 26-29 are also patentable under 35 U.S.C. §102(e) over Bennett and Levit-Gurevich.

Applicants further submit that Bennett and Levit-Gurevich do not teach or suggest identifying an IO instruction, and scanning an instruction stream to determine whether additional IO instructions are present within an extent of instructions in the instruction stream.

On the contrary, Bennett only discloses a VMM 112 that decides whether to perform a requested operation (e.g. emulate it for the guest software, proxy the operation directly to the platform hardware 116, etc.) or deny access to the resource to facility security, reliability or other mechanisms. The VMM 112 does not identify an IO instruction, and scan an instruction stream to determine whether additional IO instructions are present within an extent of instructions in the instruction stream (see Bennett paragraph [0020]).

Furthermore, Levit-Gurevich only discloses a block 326 that determines if a Virtualization Event is a complex event or not. If the Virtualization Event is not complex, then the block 328 checks if the exit Event was due to the simulated processor end of quota. If the block 238 determines that the answer is no, then control is passed to block 330 which executes code to perform a virtualization operation to handle the Virtualization Event with the DEX Monitor 212. The block 330 may perform the simulation needed for handling a non-complex event, which does not have to be sent to the Full Platform Simulator 210. The block 330 then passes control back to the block 314 for Guest state virtualization. If, he answer at the block 328 is yes, then a block 332 switches the DEX Monitor 212 to control of the next Vmx+1. Control is passed to block 314 (see Levit-Gurevich paragraph [0035] and Figure 3). Levit-Gurevich does not teach or suggest identifying an IO instruction, and scanning an instruction stream to determine whether additional IO instructions are present within an extent of instructions in the instruction stream.

In contrast, claim 2 states

> The method of Claim 1, further comprising:
> identifying an IO instruction; and
> scanning the instruction stream to determine whether additional IO instructions are present within an extent of instructions in the instruction stream.

(Claim 2) (Emphasis added).

Claim 19 includes similar limitations. Given that claims 3-9 depend directly or indirectly from claim 2, and claims 20-22, and 26-29 depend directly or indirectly from claim 19, it is likewise submitted that claims 3-9, 20-22, and 26-29 are also patentable under 35 U.S.C. §102(e) over Bennett and Levit-Gurevich.

In view of the amendments and arguments set forth herein, it is respectfully submitted that the applicable rejections and have been overcome. Accordingly, it is respectfully submitted that claims 1-9, 18-22, and 26-30 should be found to be in condition for allowance.

If any additional fee is required, please charge Deposit Account No. 50-1624.

Respectfully submitted,

Lawrence Cho Attorney at Law
C/O PortfolioIP
P.O. Box 52050
Minneapolis, MN 55402
Telephone Number 217-377-2500

Date February 2, 2006

By _____
Lawrence M. Cho
Reg. No. 39,942

CERTIFICATE UNDER 37 CFR 1.8: The undersigned hereby certifies that this correspondence is being deposited with the United States Postal Service with sufficient postage as first class mail, in an envelope addressed to: Mail Stop Amendment, Commissioner of Patents, P.O.Box 1450, Alexandria, VA 22313-1450, on this 2 day of February 2006.

Kyrstin Ryan
Name

Kyrstin Ryan
Signature